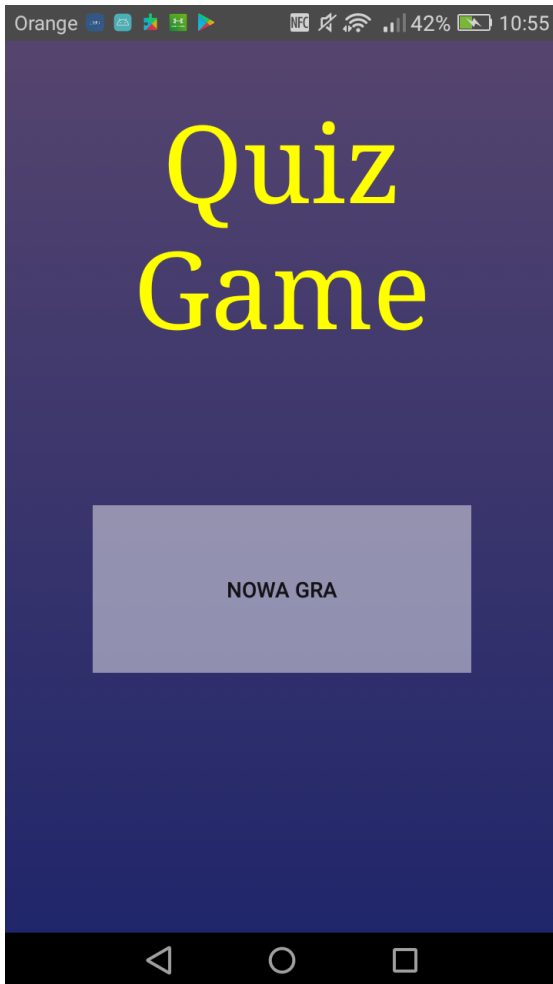


# „Quiz Game”

Autor: Adam Włosek

## 1. Wygląd aplikacji



## 2. Wstęp Teoretyczny

„Quiz Game” to gra w której gracz ma za zadanie odgadnąć widniejącą flagę danego państwa, sprawdzając czy dana litera zawiera się w nazwie danego kraju. Za każdą rozszyfrowaną nazwę flagi jest przyznawane 1 pkt. Gracz ma tylko 3 próby, gdzie w chwili wykorzystania wszystkich z nich – gra kończy się podliczając ilość punktów za każdą odgadniętą nazwę.

Motywacją do stworzenia tej aplikacji była chęć połączenia przyjemnego z pożytecznym – czyli stworzenia ciekawej gry, której wykonanie pozwoli wykorzystać mój obecny potencjał a także, pozwoli na użycie podstawowych jak i tych zaawansowanych narzędzi przy projektowaniu wyglądu aplikacji oraz silnika gry.

### 3. Wykorzystane rozwiązania i dokumentacja projektu

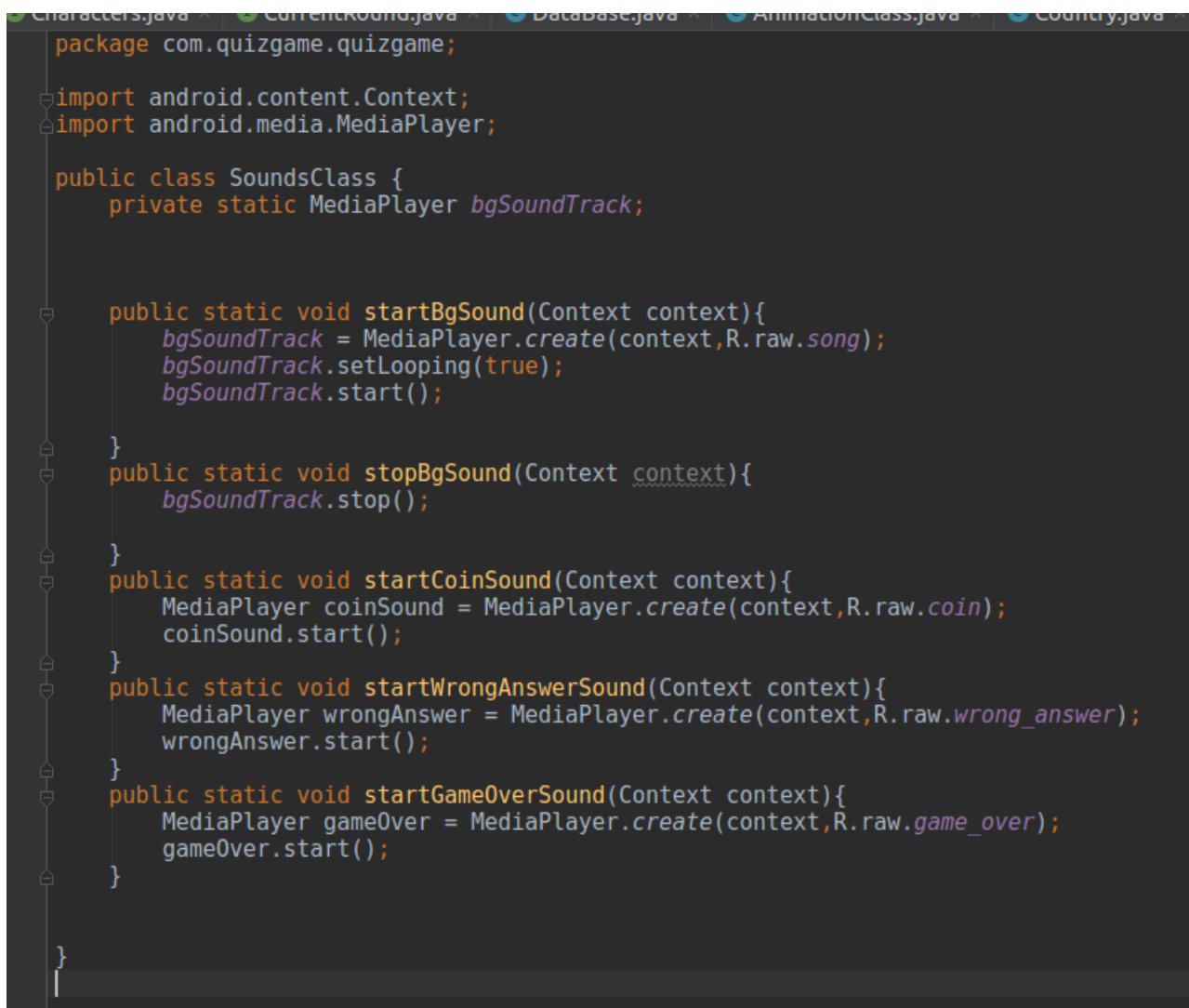
Kod programu zawiera w sobie wiele funkcjonalnych narzędzi i bibliotek wykorzystanych w aplikacji. Słowniki, Listy, interfejsy ,które usprawniają aplikację wpływając na lepszą czytelność kodu źródłowego i wydajność. W projekcie umieściłem ,także kilka sampli dźwiękowych obsługiwanych przez obiekty klasy MediaPlayer m.in. muzykę odtwarzaną w tle ,która uatrakcyjnia jakość rozgrywki. Jeżeli chodzi o tło aplikacji, zastosowałem animowany gradient zmieniający kolory, obsługiwany przez obiekty klasy AnimationDrawable, a także obiekty klasy ValueAnimator odpowiedzialne za zmianę koloru przycisków podczas kliknięcia.

Sam projekt jest podzielony na 9 klas i 2 interfejsy:

- Klasa AnimationClass
- Klasa Country
- Klasa DataBase
- Klasa Game Engine
- Klasa

Np	Typ	Nazwa	Funkcjonalność
1	Class	AnimationClass	Obsługuje animację zmiany backgroundu aplikacji i koloru przycisków
2	Interface	Characters	Zawiera słownik liter alfabetu, i iteratory liter
3	Class	Country	Reprezentuje obiekty państw. Zawiera id kraju oraz jego nazwę
4	Interface	CurrentRound	Odpowiada za obecną rozgrywkę, sprawdza występowanie litery w słowie, wypełnia pole
5	Class	DataBase	Baza danych – przechowuje obiekty Country oraz identyfikatory Flag
6	Class	GameEngine	Silnik gry implementujący interfejsy Characters, CurrentRound oraz zawierający w sobie obiekty DataBase
7	Class	MainActivity	Ekran startowy aplikacji
8	Class	NewGame	Ekran rozgrywki
9	Class	Scores	Ekran uzyskanego wyniku
10	Class	SoundsClass	Klasa odpowiedzialna za oprawę dźwiękową aplikacji

## 4. Wybrane fragmenty implementacji

The image is a screenshot of an IDE window showing the implementation of the SoundClass. The package is com.quizgame.quizgame. It imports Context and MediaPlayer from android. The class SoundClass has a private static MediaPlayer bgSoundTrack. It contains five public static methods: startBgSound, stopBgSound, startCoinSound, startWrongAnswerSound, and startGameOverSound. Each method creates a MediaPlayer object, sets its context and resource ID, and starts it. The bgSoundTrack is set to loop and is not destroyed. The other methods create local objects that are destroyed when the method returns.

```
package com.quizgame.quizgame;

import android.content.Context;
import android.media.MediaPlayer;

public class SoundsClass {
    private static MediaPlayer bgSoundTrack;

    public static void startBgSound(Context context){
        bgSoundTrack = MediaPlayer.create(context,R.raw.song);
        bgSoundTrack.setLooping(true);
        bgSoundTrack.start();
    }

    public static void stopBgSound(Context context){
        bgSoundTrack.stop();
    }

    public static void startCoinSound(Context context){
        MediaPlayer coinSound = MediaPlayer.create(context,R.raw.coin);
        coinSound.start();
    }

    public static void startWrongAnswerSound(Context context){
        MediaPlayer wrongAnswer = MediaPlayer.create(context,R.raw.wrong_answer);
        wrongAnswer.start();
    }

    public static void startGameOverSound(Context context){
        MediaPlayer gameOver = MediaPlayer.create(context,R.raw.game_over);
        gameOver.start();
    }
}
```

Ilustracja 1: Implementacja klasy SoundClass

Ilustracja 1 przedstawia klasę SoundClass zawierającą statyczne pole klasy MediaPlayer oraz jej metody. Dzięki takiemu rozwiązaniu możliwa jest łatwa obsługa odtwarzanej muzyki. Metoda startBgSound tworzy nowy obiekt klasy MediaPlayer, konstruktor zaś przyjmuje wartość odpowiedzialną za lokalizację ścieżki dźwiękowej. W tym wypadku jest to R.raw.song (W folderze raw zapisywane są wszystkie dźwięki). Analogicznie pozostałe metody działają podobnie z tą różnicą że obiekt po wyjściu z metody jest od razu usuwany (z racji że jest on nieużywany) w przeciwieństwie do obiektu bgSoundTrack ,który musi istnieć cały czas tak by muzyka była odtwarzana bez przerwy.

```

1 package com.quizgame.quizgame;
2 import android.animation.ArgbEvaluator;
3 import android.animation.ObjectAnimator;
4 import android.animation.ValueAnimator;
5 import android.graphics.drawable.AnimationDrawable;
6 import android.view.View;
7 import android.widget.RelativeLayout;
8
9 public class AnimationClass {
10
11     private RelativeLayout myLayout;
12     AnimationDrawable animationDrawable;
13
14     @ public AnimationClass(RelativeLayout myLayout,int enterFadeDuration, int exitFadeDuration) {
15         this.myLayout = myLayout;
16         this.animationDrawable = (AnimationDrawable) myLayout.getBackground();
17         this.animationDrawable.setEnterFadeDuration(enterFadeDuration);
18         this.animationDrawable.setExitFadeDuration(exitFadeDuration);
19         this.animationDrawable.start();
20     }
21
22     public void startColorAnimation(View view, int colorStart, int colorEnd,int duration){
23         ValueAnimator colorAnim = ObjectAnimator.ofInt(view, propertyName: "backgroundColor",
24             colorStart,colorEnd);
25         colorAnim.setDuration(duration);
26         colorAnim.setEvaluator(new ArgbEvaluator());
27         colorAnim.setRepeatCount(0);
28         colorAnim.setRepeatMode(ValueAnimator.RESTART);
29         colorAnim.start();
30     }
31 }
32

```

Ilustracja 2: Implementacja klasy AnimationClass

Ilustracja 2 przedstawia implementację klasy AnimationClass zawierającą w sobie rozbudowany konstruktor. Obiekt RelativeLayout przekazywany w konstruktorze dotyczy layoutu na którym ma zostać zastosowana animacja tła. Zmienne enterFadeDuration i exitFadeDuration określają tempo przejścia animacji.

Metoda startColorAnimation zmienia kolor obiektu przekazanego jako View. W tym wypadku przekazałem obiekty Button po których naciśnięciu zmienia się ich kolor. Zmienna colorStart odpowiada za zmianę koloru po kliknięciu przycisk, (przechowuje ona wartości hexadecymalne odpowiedzialne za kolor) ,a colorEnd określa jaki ma być kolor końcowy przycisku. Prędkość przemijania koloru określa zmienna duration. Metoda startColorAnimation jest wykonywana w metodzie przycisku (OnClickListener) , dzięki czemu po naciśnięciu na przycisk, zmienia on swoją barwę.

## 5. Podsumowanie

Projekt który wykonałem nie stwarzał sam w sobie większych problemów. Implementacja silnika gry ,tak by sprawdzał właściwą odpowiedź i umieszczał ją w przypadku gdy jest poprawna, oraz odświeżał pola zajęła dużo czasu, tak by opracować wydajny, stabilny i sprawny algorytm. Stworzenie planu aplikacji, rozpisanie klas i ich funkcjonalności bardzo ułatwiło mi uporanie się z dużą ilością kodu. Podzielenie kodu na klasy i interfejsy, zwiększyło jego czytelność. Wystąpiły również drobne problemy wynikające z braku wiedzy na temat importowanych klas. Mowa tutaj o klasie odpowiedzialnej za animację. Po dokładnej analizie dokumentacji i zrozumieniu metod, problemy zniknęły.